VisFlow: Adaptive Content-Aware Video Analytics on Collaborative Cameras

Yuting Yan†, Sheng Zhang*†, Xiaokun Wang†, Ning Chen†, Yu Chen†, Yu Liang‡, Mingjun Xiao§, Sanglu Lu†
† State Key Lab. for Novel Software Technology, Nanjing University, P.R. China
‡ School of Computer and Electronic Information and the School of AI, Nanjing Normal University, P.R. China
§ School of Computer Science and Technology, University of Science and Technology of China, P.R. China
Email: yuting.yan@smail.nju.edu.cn, sheng@nju.edu.cn, {652023330016, ningc, yu.chen}@smail.nju.edu.cn, liangyu@njnu.edu.cn, xiaomj@ustc.edu.cn, sanglu@nju.edu.cn

IEEE INFOCOM 2024 - IEEE Conference on Computer Communications | 979-8-3503-8350-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/INFOCOM52122.2024.10621148

***Abstract*—There is an increasing demand for analyzing live surveillance video streams via large-scale camera networks, particularly for applications in public safety and smart cities. To address the conflict between resource-intensive detection models and limited capabilities of cameras, a detection-with-tracking framework has gained prominence. However, since trackers are vulnerable to occlusions and new object appearances, frequent detections are required to calibrate the results, leading to varying detection demands that depends on video content. Consequently, we propose a mechanism for content-aware analytics on collaborative cameras, denoted as VisFlow, to increase the quality of detections and achieve the latency requirement by fully utilizing camera resources. We formulate such a problem as a non-linear, integer program with a long-term perspective, aimed at maximizing detection accuracy. An online mechanism, underpinned by a queue-based algorithm and randomized rounding, is then devised to dynamically orchestrate detection workloads among cameras, thus adapting to fluctuating detection demands. Via rigorous proof, both dynamic regret regarding overall accuracy and the transmission budget are ensured in the long run. The testbed experiments on Jetson Kits demonstrate that VisFlow improves accuracy by 18.3% over the baselines.**

I. Introduction

The surge in surveillance needs has led to the deployment of cameras across various settings like factories, campuses, and crossroads [1]–[3]. These cameras produce a vast amount of video data that constantly requires meticulous analysis. However, offloading a large volume of frames to a remote cloud server is unsuitable for real-time video analytics due to high latency and heavy transmissions [4]. Meanwhile, the resources available in devices connected to the cameras are usually limited [5]. Hence, a new framework combining detection and tracking techniques has been widely used [6], [7]. It conducts lightweight object tracking continuously, while computing-intensive detections are performed at regular intervals to calibrate the results and alleviate long-term tracking drift [8]. However, the dynamic nature of video content, such as swift object movements and occlusion, challenges the performance of tracker, potentially leading to increased tracking drift and cumulative accuracy loss [9], [10]. This requires more frequent detection calibrations to maintain high precision, thereby resulting in increased detection demands. Therefore, in a system with collaborative cameras, more resources should be provisioned for the cameras that need additional attention.

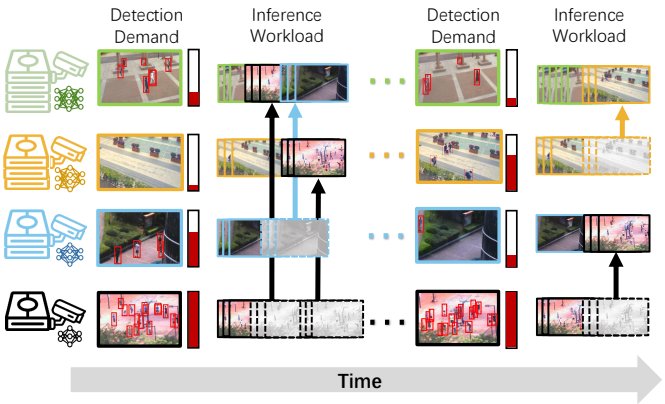


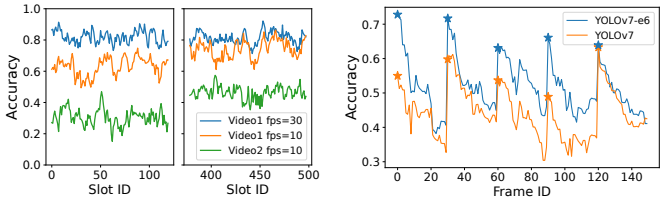
Fig. 1: An illustration of content-aware analytics. The devices with high detection demands offload frames to spare devices.

However, optimally conducting content-aware detections by workload balancing on collaborative cameras, as shown in Fig. 1, with varying video content, faces challenges as follows:

First, the detection demands of live video streams cannot be known in advance. In live video stream, the ever-changing content often causes object trackers to fail due to occlusions and the appearance of new objects, known as tracking drift [11]. Consequently, detection demand, which hinges on the current tracking performance, becomes unpredictable. For example, when using optical flow [12] to track pedestrians for 30 frames, the tracking accuracy drops by 73% in heavily crowded scenes, compared to just a 15% decrease with fewer people. Further, to maintain an accuracy of 0.9, it requires six detections to correct tracking drift in crowded scenes, but only two in less crowded ones. Thus, proactive resource provisioning for inference is the key to effectively managing vision shift among cameras.

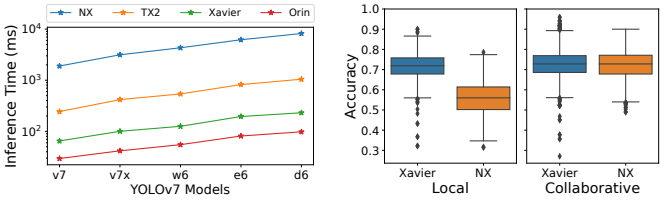
Second, the tracking accuracy relies on the previous detection quality, which should be considered jointly. Prior research has focused on adapting detection frequency and model selection in video analytics, treating each factor as independently affecting accuracy [13], [14]. However, this approach is not appropriate for the detection-with-tracking framework. A low detection frequency leads to prolonged tracking, exacerbating tracking drift. Moreover, since tracking is performed based on the former frame's result, low-quality detections lead to inaccurate tracking. For instance, using the same tracker, we

*The Corresponding Author is Sheng Zhang (sheng@nju.edu.cn). This work was supported in part by NSFC (62202233, 61832008), Double Innovation Plan of Jiangsu Province (JSSCBS20220409), and Collaborative Innovation Center of Novel Software Technology and Industrialization.



(a) Varying Video Content (b) Tracking Drift over Time

Fig. 2: Factors on Unpredictable Detection Demand



(a) Latency on Devices (b) Local v.s. Collaborative

Fig. 3: Collaborative Analytics on Heterogeneous Cameras

correct the result every 5 frames by a large model YOLOv7-e6e and a lightweight model YOLOv7 [15], respectively, and get tracking accuracy as 0.91 and 0.64.

Third, the provisioning of limited resources among heterogeneous cameras should also carefully consider both accuracy and latency. As different devices have varying computing resources, they can host models of different sizes. Larger models typically provide higher accuracy, but the inference latency is influenced by both the model size and the device capability. The tracking quality relies both on detection quality and detection frequency, facing a tradeoff on model size. Additionally, the fluctuating network environment among devices also impacts the end-to-end latency. Thus, to carefully ensure both accuracy and latency, the devices need to orchestrate detection frequency and model selection. Further, network status should be considered when offloading frames among devices.

Existing research falls insufficient for handling all the above challenges. Some works [6], [7], [16] resort to the detection-with-tracking framework, but fail to fully use resources of multiple cameras. Others [17]–[20] study analytics with distributed cameras, but ignore the tracking technique on the camera side. And the rest [21]–[23] adjust analysis configuration adaptively but fail to consider the effect of detection quality on tracking.

In this paper, we investigate content-aware video analytics on heterogeneous collaborative cameras. More specifically, for those cameras experiencing significant tracking drift and high detection demands, we enhance detection frequency and quality by offloading frames to spare devices and employing larger models. To cope with the dynamic video content, we adaptively adjust the offloading strategy within the constraint of limited resources and the need to meet latency requirement.

To tackle the problem, we formulate this scenario as a long-term, non-linear integer program aimed at maximizing the cumulative accuracy across all cameras over time. To deal with the unpredictable environments like video content and network status, we have developed a polynomial-time queue-based online learning algorithm. This algorithm learns from past analytics's accuracy, thereby facilitating improved decisions for the current time slot. We also factor in device energy consumption to manage long-term electricity costs. The proposed algorithm comprises two main components. The first component solves the decision regarding the inference frame number assigned to each device in the real domain, using feedback from previous time slots, such as tracking drift and network status. We use queues to monitor and control the cumulative transmission and energy cost, and to guide the decision-making over time. The second component further refines the decisions into a practical offloading strategy, ensuring integer decisions and adherence to predefined constraints. Via rigorous proof, both the dynamic regret in terms of accuracy and the dynamic violation regarding transmission and energy costs grow only sublinearly compared to the optimal scenario.

We implement a testbed upon four heterogeneous Jetson Developer Kits and also verify the performance of large-scale cameras for real-time video analytics. We use a series of models from YOLOv7 for detections, and the optical flow approach for tracking, which are both typical and widely used techniques. The testbed results show that our algorithm achieves up to an 18.3% improvement in accuracy compared to state-of-the-art alternatives. Additionally, our algorithm adaptively adjusts the frame offloading strategy to facilitate content-aware analytics across cameras with varying network conditions.

II. System Models and Problem Formulation

A. Preliminary Case Study: Analytics on Distributed Cameras

Unpredictable Detection Demands: Due to limited computing resources on smart cameras, it is hard to ensure real-time performance for video analytics. Hence, there is a new trend to combine detection with object tracking. Specifically, frames are continuously processed by a lightweight tracker, such as optical flow or KCF [24]. This process generates new bounding boxes for objects by using the result from the former frame and analyzing the deviation between consecutive frames. In addition, as long-term tracking leads to cumulative accuracy loss, necessary detections are frequently performed to correct the bounding boxes obtained by tracker. Unfortunately, the video content greatly affects the performance of the tracker. For example, factors like occlusions and changes in object appearance can lead to more tracking drift [11]. Thus, the detection demand also varies with the video content.

In Fig. 2(a), we test the analytics quality on two videos using YOLOv7-w6 with different detection frequencies. The two videos are both downloaded from street surveillance camera lives on YouTube [25], [26], with 30fps. We set per slot as one second, and use the average accuracy of 30 frames as the slot accuracy. The result depicts that for Video1, when the detection frequency is set to 30fps and then to 10fps, the average accuracy gap shifts from 0.19 to 0.05, corresponding to the changes in video content. Intuitively, a relatively low detection frequency suffices when the video content is clearer, whereas more detections is needed when the content is chaotic. Further, with detection frequency both set as 10fps, the accuracy of Video1 is greater than Video2 by 0.3 on average. Thus, even with the same detection frequency, accuracy varies across

different videos. This underscores the need for careful consideration of the detection demands for each camera.

Fig. 2(b) shows the accuracy of detection-with-tracking framework over time. We use YOLOv7 or YOLOv7-e6 as the detector and optical flow as the tracker. The detection is triggered once per second. The result illustrates that the long-term tracking drift leads to a significant and cumulative accuracy drop. Specifically, tracking accuracy decreases 27%-44% after continuously tracking 29 frames, and the decrease varies from video content. Consequently, increased detection frequency is necessary as tracking accuracy drops fast. Moreover, the initial detection quality also affects the onset of tracking accuracy. As using the larger model YOLOv7-e6, the average detection accuracy is 20% higher than with YOLOv7, and the tracking accuracy is superior by 17.1%. Therefore, the impact of model selection on tracking should also be considered.

Content-Aware Analytics on Heterogeneous Cameras: Nowadays plenty of cameras are deployed at campus or crossroads, which are often connected to heterogeneous devices with varying resources. Owing to factors like purchasing costs, most devices in a distributed camera system tend to be weak, with only a few having adequate computing resources. Unfortunately, these weaker devices often become overloaded, struggling to handle varying detection demands and maintain high-quality analytics. Thus, conducting content-aware detections and offloading frame inference tasks from overloaded to underutilized devices could be a helpful solution.

Fig. 3(a) shows the inference time cost of models on heterogeneous devices. For our testbed, we utilize four widely used devices: Jetson Xavier NX, TX2, AGX Xavier, and AGX Orin, along with a series of YOLOv7 variants. The test image resolution is set the same as the official instructions, i.e., 640x480 for YOLOv7 and YOLOv7x, and 1280x720 for the others. The result shows the great performance gap among devices. For instance, it takes 1,896 ms to infer a 480p frame using YOLOv7 on NX, while it only takes 29.45 ms on Orin. That is, while YOLOv7 runs at less than 1fps on the NX, it can achieve 33fps on the Orin. Thus, a powerful device has the potential to take on tasks from several weaker devices.

Fig. 3(b) further depicts the accuracy comparison between the local inference approach and content-aware collaborative inference approach. The testbed utilizes a Xavier and an NX device, each running a detection-with-tracking framework with YOLOv7x and optical flow. Under the local inference approach, each device performs as many as possible inferences locally, but the NX achieves only an average accuracy of 0.56 due to high inference latency. Conversely, with collaborative inference approach, the workload is redistributed between devices based on video content and detection demands. As a result, the average accuracy on the NX improves to 0.72, and the Xavier yields more results with high accuracy.

Insights: From the studies above, we conclude that: (1) The tracking drift and detection demand of each camera fluctuates as the video content changes. (2) Conducting content-aware detections and balancing workloads in response to varying detection demand across cameras enhances analysis quality.

B. System Settings and Models

We model our multi-camera collaborative video analytics system as follows. The system contains a group of smart camera devices, each denoted by $\mathcal{N} = \{1, 2, \ldots, N\}$, connected via a wired backhaul. The camera devices generate video streaming continuously, and we divide time into a series of slots $\mathcal{T} = \{1, \ldots, T\}$. In each slot, F new frames are captured by each camera, and the resolution varies from cameras. Each device is deployed with an object detection model, which varies among devices according to capability. Since the devices have different resource capacities, we consider there are N_0 strong devices with sufficient GPU resource, which are able to accept frame processing task from other devices, denoted by $\mathcal{N}_0 = \{1, \ldots, N_0\}$. And the other N_1 devices are weak, which can only offload tasks to strong devices or process frames locally, denoted by $\mathcal{N}_1 = \{N_0 + 1, \ldots, N\}$. It matches most realistic cases that usually only small part of devices are much better than others.

Control Decisions: We introduce the control decision to facilitate the modeling. In time slot t, we need to decide how many frames to offload to other devices or to infer locally for each device. For device i, we use $\mathcal{N}_i$ to denote the available devices to offload. For strong device i, $\mathcal{N}_i = \mathcal{N}_0$; and for weak device i, $\mathcal{N}_i = \mathcal{N}_0 \cup \{i\}$ (strong devices and itself). Thus, we use $x_{i,j}^t \in [F], \forall i \in \mathcal{N}, j \in \mathcal{N}_i, t \in \mathcal{T}$ to denote how many frames of device i should be processed on device j in slot t, where $[F] = \{0, 1, \ldots, F\}$. Our goal is to decide proper inference frames for each device and assign the frames among devices to fully utilize resources. From the control decision, in slot t, the number of inferred frames of device i can be measured as $F_i^t = \sum_{j \in \mathcal{N}_i} x_{i,j}^t$. Then we uniformly divide the F frames into F_i^t windows, make detection for the first frame and use the result to track the other frames in each window.

Detection Accuracy: For each frame, we use the IoU (Intersection over Union) between the bounding boxes and the ground truth to measure the accuracy. Under the detection-with-tracking framework, if a frame is inferred, the bounding boxes are derived from detection. Otherwise, the result has to be updated by tracking. Hence, we model the accuracy of detection frames and tracking frames respectively. For detections, we use a_{i,m_j} to denote the detection accuracy of video streaming from device i using the model m_j on device j. Thus, for device i, the total detection accuracy for all the inferred frames can be measured as

$$A_{i,t}^{det} = \sum\nolimits_{j \in \mathcal{N}_i} a_{i,m_j} x_{i,j}^t. \tag{1}$$

For tracking frames, according to the preliminary study, tracking accuracy relies on the initial detection accuracy and tracking drift occurs as time goes. Thus, we denote $\phi_i^t \in [0, 1]$ the tracking drift factor of device i in slot t, which measures the accuracy loss compared to the former frame. Then, the total tracking accuracy for device i can be measured as

$$\begin{aligned} A_{i,t}^{track} &= \alpha_i^t \cdot F_i^t \cdot (\phi_i^t + (\phi_i^t)^2 + \cdots + (\phi_i^t)^{F/F_i^t - 1}) \\ &= \alpha_i^t \cdot F_i^t \cdot \frac{\phi_i^t (1 - (\phi_i^t)^{F/F_i^t - 1})}{1 - \phi_i^t}, \end{aligned} \tag{2}$$

where α_i^t denotes the average initial detection accuracy of each tracking window, F_i^t equals to the window nuber, and there are $F/F_i^t - 1$ frames to track in each window. Hence, the overall accuracy of device i in slot t is $A_{i,t} = A_{i,t}^{det} + A_{i,t}^{track}$.

Inference Latency: For each slot t, each device needs to infer frames from itself, and strong devices need to infer frames offloaded from other devices additionally. Since the tracker such as optical flow is lightweight and costs only CPU resources, which is relatively sufficient to track frames in real-time, while the inference costs limited GPU resource of devices, we consider the inference latency as the bottleneck. We use $T_{i,j}$ to denote the inference time of device j using m_j to process a frame from device i. Thus, the inference latency for device i can be measured as

$$L_{i,t} = T_{i,i} \cdot x_{i,i}^t + \mathbb{I}[i \in \mathcal{N}_0] \sum\nolimits_{j \in \mathcal{N} \setminus i} T_{j,i} \cdot x_{j,i}^t, \quad (3)$$

where $\mathbb{I}[i \in \mathcal{N}_0]$ indicates whether device i is strong.

Transmission Budget: Since the network status and bandwidth among devices fluctuate over time, the transmission of each device has a budget. Since the frame data size is much greater than the labels, we only consider the data size d_i of one frame from device i. For weak devices, the transmission only comes from offloading frames, but for strong devices, it also comes from receiving frames. Thus, for any device i in slot t, the total transmission $D_{i,t}$ can be measured by adding the data it sends and receives:

$$D_{i,t} = \sum\nolimits_{j \in \mathcal{N}_i \setminus i} d_i x_{i,j}^t + \mathbb{I}[i \in \mathcal{N}_0] \sum\nolimits_{j \in \mathcal{N} \setminus i} d_j x_{j,i}^t, \quad (4)$$

where $x_{i,j}^t$ and $x_{j,i}^t$ denote how many frames device i sends to and receives from j in slot t, respectively.

Energy Consumption: Since electricity cost is one of the primary costs when maintaining the video analytics system in a long period, we take energy efficiency into consideration. Since the energy consumed by transmission is much lower than the inference using GPU in our testbed [27], we mainly consider the inference energy. We use γ_i to denote the energy cost for device i to make inference in a unit time, thus the energy cost of device i in slot t can be measured as

$$E_{i,t} = \gamma_i \cdot L_{i,t}. \quad (5)$$

C. Problem Formulation and Challenges

Control Problem $\mathcal{P}^G$: With the above system model, we propose the following control problem, aiming at maximizing the overall accuracy of all the devices with constrained inference latency, transmission budget and energy consumption.

$$\begin{aligned}
\mathcal{P}^G: \max \quad & \sum\nolimits_t \sum\nolimits_{i \in \mathcal{N}} A_{i,t} && (6)\\
s.t. \quad & L_{i,t} \le T_{slot}, \forall i \in \mathcal{N}, t \in \mathcal{T} && (7)\\
& \frac{1}{T} \sum\nolimits_t D_{i,t} \le B_{i,t}, \forall i \in \mathcal{N} && (8)\\
& \frac{1}{T} \sum\nolimits_t \sum\nolimits_{i \in \mathcal{N}} E_{i,t} \le E_{max} && (9)\\
& \sum\nolimits_{j \in \mathcal{N}_i} x_{i,j}^t \ge 1, \forall i \in \mathcal{N}, t \in \mathcal{T} && (10)\\
& F_i^t \le F, \forall i \in \mathcal{N}, t \in \mathcal{T} && (11)\\
var. \quad & x_{i,j}^t \in [F], \forall i \in \mathcal{N}, j \in \mathcal{N}_i, t \in \mathcal{T} && (12)
\end{aligned}$$

Constraint (7) ensures that the processing time of each device does not exceed the total time of slot T_{slot}, i.e., the inference tasks should not overwhelm the device capability. Constraint (8) ensures that for each device i, the total transmission should not exceed the transmission budget $B_{i,t}$. Since the transmission budget relies on network status and cannot be known in advance, satisfying the budget in each time slot t is impossible. Thus, we aim to satisfy the constraint in the long term. Constraint (9) restricts the average energy consumption is no more than E_{max} in the long term, which controls the electricity cost of the system. Constraint (10) ensures that each device should make inference for at least one frame in each slot, which avoids long-term tracking drift and accuracy loss. Constraint (11) ensures the total inference frame does not exceed the total frame num per slot for each device. Constraint (12) restricts the domain of our decision variables.

Control Problem $\mathcal{P}^D$ with Observable Inputs: Since there is no oracle model, the ground truth cannot be known in the analytic period. Therefore, we have to use our detection results as an approximate substitute for the ground truth. For example, tracking drift factor $\hat{\phi}_i^t$ need to be updated using our detection results, and the details are in Section III. Then, we reformulate problem $\mathcal{P}^G$ with observable inputs:

$$\begin{aligned}
\mathcal{P}^D: \max \quad & \sum\nolimits_t \sum\nolimits_{i \in \mathcal{N}} \hat{A}_{i,t} \\
s.t. \quad & \text{Constraints (7)} \sim \text{(12)}, && (13)
\end{aligned}$$

where $\hat{A}_{i,t}$ denotes the accuracy from observable inputs.

Problem Challenges: However, it is still hard to solve the problem $\mathcal{P}^D$ with the observable inputs. First, as $x_{i,j}^t \in [F]$, the problem belongs a to non-linear integer program, which is no easier than an NP-Complete [28] problem. Second, as the system is running, the arrival of frames is in an online manner, which leads to extra uncertainty. It's non-trivial to make frame inference offloading decisions in an ever-changing environment considering accuracy, inference latency, transmission limit, and energy cost meanwhile.

III. Online Algorithm Design

To handle the challenges, we use a virtual queue to measure the violation of the long-term constraint and decouple the problem into a series of subproblems per epoch. Fig. 4 depicts the relationship between all the problems and the solutions, which illustrates how we solve the problem step by step.

A. Notations and Transformations

Transformation of problem: For simplicity, we reformulate the problem $\mathcal{P}^D$ into the canonical form:

$$\begin{aligned}
&\min \sum_{t=1}^{T} f_t(\boldsymbol{x}_t) \quad s.t. \sum_{t=1}^{T} \boldsymbol{g}_t(\boldsymbol{x}_t) \preceq \mathbf{0}; \forall t: \boldsymbol{h}(\boldsymbol{x}_t) \preceq \mathbf{0}; \boldsymbol{x}_t \in \mathcal{X},\\
&\forall t: f_t(\boldsymbol{x}_t) \coloneqq -\sum\nolimits_{T \in \mathcal{T}, i \in \mathcal{N}} \hat{A}_{i,t},\\
&\forall t: \boldsymbol{g}_t(\boldsymbol{x}_t) \coloneqq [\forall i\colon D_{i,t} - B_{i,t}; \sum\nolimits_{i \in \mathcal{N}} E_{i,t} \le E_{max}],\\
&\forall t: \boldsymbol{h}(\boldsymbol{x}_t) \coloneqq [\forall i\colon L_{i,t} - T_{slot}; 1 - \sum\nolimits_{j \in \mathcal{N}_i} x_{i,j}^t; F_i^t - F],
\end{aligned}$$

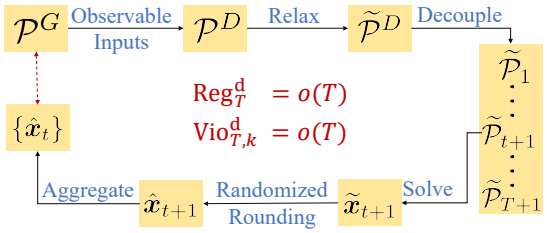


Fig. 4: The Roadmap for Proposed Algorithm

where $\boldsymbol{x}_t$ denotes the aggregation of $\{x_{i,j}^t\}$ in slot t, whose integral domain is denoted by $\mathcal{X} = [F]^{N_0 \times N+N_1}$. Note that taking the opposite of original objective as f_t transforms $\mathcal{P}^D$ into a minimization problem, which facilitates our analysis later. Also, we define the corresponding minimization objective of $\mathcal{P}^G$ in slot t as $f_t^G = -\sum_{i \in \mathcal{N}} A_{i,t}$.

Relaxation: To deal with the NP-hardness of $\mathcal{P}^D$, we relax $\mathcal{P}^D$ to the real domain. We define $\widetilde{\boldsymbol{x}}_t$ as the fraction version of $\boldsymbol{x}_t$ and its domain is $\widetilde{\mathcal{X}} = [0, F]^{N_0 \times N+N_1}$. Thus, we aim to solve the problem $\widetilde{\mathcal{P}}^D$ in real domain:

$$\begin{aligned} \min \quad & \sum_{t=1}^{T} f_t(\widetilde{\boldsymbol{x}}_t), \\ s.t. \quad & \sum_{t=1}^{T} \boldsymbol{g}_t(\widetilde{\boldsymbol{x}}_t) \preceq \mathbf{0}; \forall t: \boldsymbol{h}(\widetilde{\boldsymbol{x}}_t) \preceq \mathbf{0}, \widetilde{\boldsymbol{x}}_t \in \widetilde{\mathcal{X}}. \end{aligned}$$

Lemma 1. *$\widetilde{\mathcal{P}}^D$ is a convex optimization problem.*

Proof. First, f_t is the sum of a linear function $-A_{i,t}^{det}$, and $-A_{i,t}^{track}$. Since $A_{i,t}^{track}$ is a concave function to F_i^t with $\phi_i^t \in [0,1]$ and F_i^t is a linear function to $\boldsymbol{x}_t$, we can get $A_{i,t}^{track}$ is a concave function to $\boldsymbol{x}_t$ with concave function composition property. Hence f_t is a convex function. Note that $\boldsymbol{g}_t, \boldsymbol{h}$ are composed of linear functions and the domain of $\widetilde{\boldsymbol{x}}_t$ is a convex set. Therefore, $\widetilde{\mathcal{P}}^D$ is a convex optimization problem. □

B. Joint Adaptation Algorithm

Queue-Based Approach: Since the problem is an online long-term optimization, we propose a queue-based approach to decouple it into a series of subproblems. By solving subproblems, we minimize the objective of the original problem and avoid the long-term constraints being violated heavily.

We introduce a queue for each long-term constraint, aggregated as a vector $\boldsymbol{Q}_t$ such that:

$$\forall t, \boldsymbol{Q}_{t+1} = \max\{-\boldsymbol{g}_t(\boldsymbol{x}_{t+1}), \boldsymbol{Q}_t + \boldsymbol{g}_t(\boldsymbol{x}_{t+1})\}, \tag{14}$$

where $\boldsymbol{Q}_t = [Q_{1,t}, \ldots, Q_{N,t}, Q_t^E] \in \mathbb{R}^{N+1}$, and $\max\{\cdot\}$ means to get maximum for each dimension of vector. In each slot, $\boldsymbol{Q}_t$ updates with increments $\boldsymbol{g}_t(\boldsymbol{x}_{t+1})$ to measure the accumulative violation of long-term constraints. Specifically, the first N elements represent the violation of the transmission budget of each device, which can be caused by network fluctuation; and the last element represents the violation of energy consumption, caused by excessive inference in one slot.

Decoupled Subproblem: Considering the accuracy, violation of transmission and energy, and the regular term jointly, we optimize the subproblem $\mathcal{P}_{t+1}$ decoupled from $\widetilde{\mathcal{P}}^D$:

$$\begin{aligned} \mathcal{P}_{t+1}: \min_{\boldsymbol{x} \in \widetilde{\mathcal{X}}} & \left\{ f_t(\boldsymbol{x}) + (\boldsymbol{Q}_t + \boldsymbol{g}_{t-1}(\boldsymbol{x}_t))^{\mathsf{T}} \boldsymbol{g}_t(\boldsymbol{x}) + \mu \|\boldsymbol{x} - \boldsymbol{x}_t\|_2^2 \right\} \\ s.t. \quad & \boldsymbol{h}(\boldsymbol{x}) \preceq \mathbf{0}. \end{aligned}$$

Algorithm 1 VisFlow

Input: T_{slot}, E_{max}.

1: Initialize step size $\mu_t = 1/t$, $\boldsymbol{Q}_1 = \mathbf{0}$, $\boldsymbol{g}_0(\cdot) = \mathbf{0}$;
2: Initialize $\widetilde{\boldsymbol{x}}_1$: inferring as many frames as possible on each device locally;
3: **for** slot $t = 1, 2, \ldots, T$ **do**
// 1. round $\widetilde{\boldsymbol{x}}_t$ randomly to feasible solution $\hat{\boldsymbol{x}}_t$
4: **for** $\widetilde{x}_{i,j}^t$ in $\widetilde{\boldsymbol{x}}_t$ **do**
5: $\hat{x}_{i,j}^t = \begin{cases} \lceil \widetilde{x}_{i,j}^t \rceil, \text{with probability } \widetilde{x}_{i,j}^t - \lfloor \widetilde{x}_{i,j}^t \rfloor, \\ \lfloor \widetilde{x}_{i,j}^t \rfloor, \text{with probability } \lceil \widetilde{x}_{i,j}^t \rceil - \widetilde{x}_{i,j}^t; \end{cases}$
6: **end for**
7: Deploy the provisioning $\hat{\boldsymbol{x}}_t$ to cameras, offload frames and get detection results;
8: Update f_t and $\boldsymbol{g}_t$ using the revealed information;
// 2. solve solution for next slot and update queue
9: Obtain $\widetilde{\boldsymbol{x}}_{t+1}$ by optimizing $\mathcal{P}_{t+1}$;
10: $\boldsymbol{Q}_{t+1} = \max\{-\boldsymbol{g}_t(\widetilde{\boldsymbol{x}}_{t+1}), \boldsymbol{Q}_t + \boldsymbol{g}_t(\widetilde{\boldsymbol{x}}_{t+1})\}$;
11: **end for**

At the end of slot t, we solve $\mathcal{P}_{t+1}$ to get the fractional decision $\widetilde{\boldsymbol{x}}_{t+1}$ using the information shown in the last slot. The vector $\boldsymbol{Q}_t + \boldsymbol{g}_{t-1}(\boldsymbol{x}_t)$ measures the violation of $\boldsymbol{g}_t$ by previous decisions. As the violation increases, the optimization tends to decrease the amount of data transmission or energy consumption in the current slot to mitigate the violation. And $\mu > 0$ is the step size, controlling the distance between decisions in current slot and last slot. Note that due to Lemma 1 and $\boldsymbol{Q}_t + \boldsymbol{g}_{t-1}(\boldsymbol{x}_t) \succeq \mathbf{0}$, $\mathcal{P}_{t+1}$ is also a convex optimization problem which can be solved efficiently.

Randomized Rounding: Although we relaxed the problem into fraction form and solved $\widetilde{\boldsymbol{x}}_t$ in the real domain, the number of offloaded inference frames must be an integer. Thus, we use a randomized rounding strategy to revise $\widetilde{\boldsymbol{x}}_t$ into a feasible integer provisioning. More specifically, each element $\widetilde{x}_{i,j}^t$ in $\widetilde{\boldsymbol{x}}_t$ is rounded to $\lceil \widetilde{x}_{i,j}^t \rceil$ with the probability of $\widetilde{x}_{i,j}^t - \lfloor \widetilde{x}_{i,j}^t \rfloor$, otherwise to $\lfloor \widetilde{x}_{i,j}^t \rfloor$. We highlight that this randomizing strategy ensures $\mathbb{E}[\boldsymbol{x}_t] = \widetilde{\boldsymbol{x}}_t$, which facilitates our performance analysis later.

Online Adaptation: We propose our algorithm VisFlow in Algorithm 1, which contains two components: randomized rounding and queue-based online optimization. As the system starts running, we first set the initial value for the parameters of algorithm, as shown in lines 1-2. We set step size as $\mu_t = 1/t$ to facilitate the analysis later. Note that we manually set the inference plan in slot 1. In the first component, at the start of each epoch, we convert $\widetilde{\boldsymbol{x}}_t$ into feasible integral solution $\hat{\boldsymbol{x}}_t$ by randomized rounding, in lines 4-6. In line 7, we deploy the decision to cameras, each device transmits frames and makes inferences. In line 8, the functions of slot t are updated using the accuracy and network status observed when deploying. In lines 9-10, we get the fractional decision for the next slot using the newly observed function, and update the queue to measure the constraint violation.

Parameter Obtaining: As shown in preliminary studies and previous works [14], [29], a_{i,m_j}, $T_{i,j}$, d_i and γ_i can be

estimated offline and pre-stored as a table. After deploying the offloading decision $\widetilde{\boldsymbol{x}}_t$ in slot t, we update the tracking loss by comparing the detection result and tracking result of the same frame, and fit $\hat{\phi}_i^t$ using the accuracy difference and continuous tracking frame number. Also, α_i^t can be estimated by the average detection accuracy using a_{i,m_j} of inferred frames of device i. The transmission budget B_i can be observed during data transmission in each slot.

IV. Performance Analysis

We first introduce two metrics to measure the performance: dynamic regret [30], [31] and dynamic violation [32]. Then we prove that the dynamic regret and dynamic violation of our algorithm would be sublinear with respect to T, which means our solution sequence is no worse than the dynamic optimal solution asymptotically as time length goes to infinity.

A. Metrics of Performance

Dynamic Regret: The dynamic regret is proposed to measure the difference between the long-term objective of online decisions $\{\boldsymbol{x}_t\}$ and the dynamic benchmark. The benchmark $\{\boldsymbol{x}_t^*\}$ is the solution from optimizing the one-shot problem in each slot with an oracle that knows the corresponding inputs in advance. We use $\{\hat{\boldsymbol{x}}_t^*\}$ and $\{\widetilde{\boldsymbol{x}}_t^*\}$ to denote optimal solutions of dynamic optimization problem in integral and real domains, respectively. Then the dynamic regret in integral and real domains is defined respectively as follows:

$$\mathrm{Reg}_T^{\mathrm{d}} = \mathbb{E}[\sum\nolimits_{t=1}^{T} f_t(\hat{\boldsymbol{x}}_t)] - \sum\nolimits_{t=1}^{T} f_t^G(\hat{\boldsymbol{x}}_t^*), \tag{15}$$
$$\hat{\boldsymbol{x}}_t^* \in \arg\min\nolimits_{\boldsymbol{x}\in\mathcal{X}} f_t(\boldsymbol{x}), s.t.\ \boldsymbol{g}_t(\boldsymbol{x}) \preceq \boldsymbol{0}, \boldsymbol{h}(\boldsymbol{x}) \preceq \boldsymbol{0},$$
$$\widetilde{\mathrm{Reg}}_T^{\mathrm{d}} = \sum\nolimits_{t=1}^{T} f_t(\widetilde{\boldsymbol{x}}_t) - \sum\nolimits_{t=1}^{T} f_t^G(\widetilde{\boldsymbol{x}}_t^*), \tag{16}$$
$$\widetilde{\boldsymbol{x}}_t^* \in \arg\min\nolimits_{\boldsymbol{x}\in\widetilde{\mathcal{X}}} f_t(\boldsymbol{x}), s.t.\ \boldsymbol{g}_t(\boldsymbol{x}) \preceq \boldsymbol{0}, \boldsymbol{h}(\boldsymbol{x}) \preceq \boldsymbol{0},$$

where the expectation in Eq. (15) is introduced to deal with randomized rounding.

Dynamic Violation: To guarantee the long-term constraint is not violated much in the long run, the dynamic violation is introduced to measure the cumulative violation of the long-term constraints, incurred by online decisions. Thus, for any $k = 1, \ldots, N+1$, we consider the dynamic violation in integral and real domain as follows:

$$\mathrm{Vio}_{T,k}^{\mathrm{d}} = \mathbb{E}[\sum\nolimits_{t=1}^{T} g_{t,k}(\hat{\boldsymbol{x}}_t)], \tag{17}$$
$$\widetilde{\mathrm{Vio}}_{T,k}^{\mathrm{d}} = \sum\nolimits_{t=1}^{T} g_{t,k}(\widetilde{\boldsymbol{x}}_t), \tag{18}$$

where $g_{t,k}$ is the k-th element of long-term constraints $\boldsymbol{g}_t$. Also, we take the expectation of the constraints in $\mathrm{Vio}_{T,k}^{\mathrm{d}}$ to eliminate the impact of the randomness.

B. Regret and Violation Analysis

We aim to show the dynamic regret and dynamic violation of our algorithm are both sublinear to T. To this end, we first propose several basic assumptions and notations to facilitate our analysis.

Assumptions: We propose three assumptions as follows:

Assumption 1: The domains of the decision, objective and long-term constraint can be bounded by a constant respectively, i.e., $\forall t, \boldsymbol{x} \in \widetilde{\mathcal{X}}, \|\boldsymbol{x}\|_2 \le R, |f_t(\boldsymbol{x})| \le F, \|\boldsymbol{g}_t(\boldsymbol{x})\|_2 \le G$.

Assumption 2: $\forall t, \boldsymbol{g}_t$ is Lipschitz continuous with modulus β, i.e., $\forall \boldsymbol{x}_1, \boldsymbol{x}_2 \in \widetilde{\mathcal{X}}, \|\boldsymbol{g}_t(\boldsymbol{x}_1) - \boldsymbol{g}_t(\boldsymbol{x}_2)\|_2 \le \beta \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2$.

Assumption 3: The objective with observable inputs is not far away from ground truth, i.e., $\sum_{t=1}^{T}[f_t(\widetilde{\boldsymbol{x}}_t^*) - f_t^G(\widetilde{\boldsymbol{x}}_t^*)] \le \theta_0$.

We note that Assumption 1 bounds the domain of individual decisions in each slot and their impacts on accuracy, latency and energy cost, so that an occasional bad decision only has limited impact on overall performance. Assumption 2 is the standard Lipschitz assumption, which is common in primal-dual online algorithm analysis. Assumption 3 ensures that the optimal solution of $\mathcal{P}^D$ is approximately optimal to $\mathcal{P}^G$.

Notations: We quantify the total variations of dynamic optimal point sequence $\{\widetilde{\boldsymbol{x}}_t^*\}$ as

$$V_{\boldsymbol{x}} = \sum\nolimits_{t=1}^{T} \left\|\widetilde{\boldsymbol{x}}_{t+1}^* - \widetilde{\boldsymbol{x}}_t^*\right\|_2. \tag{19}$$

Then, we define the variation of dual optimal solution sequence $\{\boldsymbol{\lambda}_t^*\}$ as

$$V_{\boldsymbol{\lambda}} = \sum\nolimits_{t=1}^{T} \left\|\boldsymbol{\lambda}_{t+1}^* - \boldsymbol{\lambda}_t^*\right\|_2. \tag{20}$$

Furthermore, we define the total variation of accuracy f_t, transmission and energy constraint $\boldsymbol{g}_t$, and also total squared variation of $\boldsymbol{g}_t$:

$$V_f = \sum\nolimits_{t=1}^{T} \max_{\boldsymbol{x}\in\widetilde{\mathcal{X}}} \{|f_{t+1}(\boldsymbol{x}) - f_t(\boldsymbol{x})|\}, \tag{21}$$
$$\widetilde{V}_{\boldsymbol{g}} = \sum\nolimits_{t=1}^{T} \max_{\boldsymbol{x}\in\widetilde{\mathcal{X}}} \{|\boldsymbol{g}_t(\boldsymbol{x}) - \boldsymbol{g}_{t-1}(\boldsymbol{x})|\}, \tag{22}$$
$$V_{\boldsymbol{g}} = \sum\nolimits_{t=1}^{T} \max_{\boldsymbol{x}\in\widetilde{\mathcal{X}}} \left\{\|\boldsymbol{g}_t(\boldsymbol{x}) - \boldsymbol{g}_{t-1}(\boldsymbol{x})\|_2^2\right\}. \tag{23}$$

Lemma 2. *With the definition of dynamic regret and constraint violation in integral and real domains, we have* $\forall k$,

$$Reg_T^d \le \widetilde{Reg}_T^d + M\sigma_\beta, \quad Vio_{T,k}^d = \widetilde{Vio}_{T,k}^d, \tag{24}$$

where $M\sigma_\beta$ is a constant upper-bound introduced by Jensen gap [33].

Proof. See Appendix A. □

Theorem 1. *The dynamic regret and dynamic violation in integral domain are upper-bounded respectively as:*

$$Reg_T^d \le R_T, \quad Vio_{t,k}^d \le V_T, \tag{25}$$

where $R_T = V_f + V_{\boldsymbol{g}} + 4\mu R V_{\boldsymbol{x}} + C_1$, $V_T = 4V_{\boldsymbol{\lambda}} + 2\sqrt{V_f} + \sqrt{2V_{\boldsymbol{g}}} + \widetilde{V}_{\boldsymbol{g}} + \sqrt{8\mu R V_{\boldsymbol{x}}} + C_2$, and C_1, C_2 are constants.

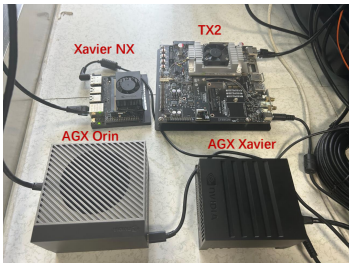
Proof. See Appendix B. □

Corollary 1. *By selecting proper step sizes in each slot, the dynamic regret and violation of our algorithm only grow sublinearly with respect to T:*

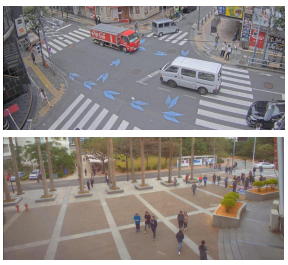
$$Reg_T^d = o(T), \quad Vio_{T,k}^d = o(T), \tag{26}$$

if $\widetilde{V}_{\boldsymbol{g}}, V_{\boldsymbol{\lambda}}$ are sublinear to T and $V_f, V_{\boldsymbol{g}}$ are $o(T^2)$ to T.

Proof. From the definition of R_T and V_T, by setting the step size as $\mu = o(T/V_{\boldsymbol{x}})$, we have $4\mu R V_{\boldsymbol{x}} = o(T)$ in R_T and $\sqrt{8\mu R V_{\boldsymbol{x}}} = o(T^{1/2})$ in V_T. Thus, each component of upper bound R_T or V_T belongs to $o(T)$, $o(T^{1/2})$ or a constant, which



(a) Devices in Our Testbed



(b) Traffic and Campus

Fig. 5: Devices Used in Testbed and Our Dataset

derives the corollary. Note that we set $\mu_t = 1/t$ in Algorithm 1 to ensure the corollary stands when $V_{\boldsymbol{x}}$ is $o(T^2)$, which fits almost all the cases. □

V. Implementation and Evaluation

A. Implementation

We implement VisFlow on four Nvidia Jetson Developer Kits as a testbed, including Xavier NX, TX2, AGX Xavier and AGX Orin, with increasing capabilities, as shown in Fig. 5(a). The devices have heterogeneous resources: For CPU, the devices have a minimum 6-core Carmel CPU and a maximum 12-core Cortex ARM CPU; For GPU, it varies from a 384-core Volta GPU to a 2048-core Ampere GPU. Thus, NX and TX2 are treated as weak devices, while Xavier and Orin are strong devices. The devices are connected to a campus local area network, and the bandwidth fluctuates around 8-13MB/s most of the time. The system is implemented in Python 3.8 environment, and the main techniques are as follows:

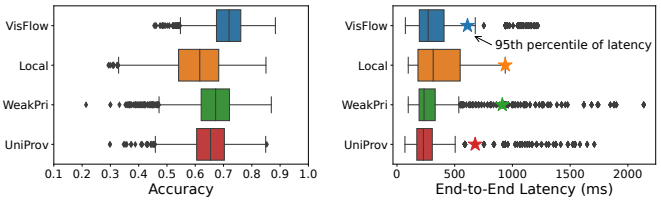
(1) Tracker: We use optical flow method in OpenCV as the tracker on devices, implemented by *calcOpticalFlowPyrLK* , which can run in real-time on each device.

(2) Detection Models: We use YOLOv7/v7x/e6/d6 as the detection model for each device respectively, and a stronger device is deployed with a larger model to provide accurate detections. Note that YOLO (and some other models like Faster-RCNN [34]) support flexible input sizes, and previous works [35] have shown that the inference time grows linearly regarding the number of pixels.

(3) Decision Scheduler: The decision is calculated by a convex programming solver *cvxpy* in Python. In each slot, we need to solve $\mathcal{P}_{t+1}$ using about 100 ms in our settings, which is deployed on Orin with sufficient CPU resources, avoiding interference with other computations. In practice, we believe it is best to use a separate device to avoid resource competition.

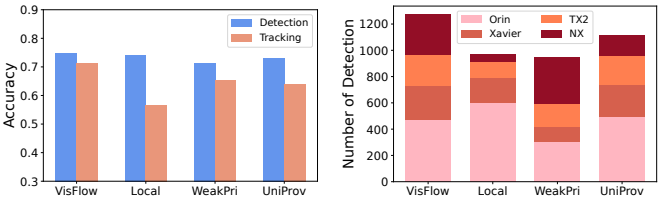
B. Data and Settings

Dataset and Parameters: We compare different approaches on surveillance videos. We build a dataset using the camera streams from Youtube [25], [26], [36], [37] and PANDA dataset [38], covering both traffic and campus scenes, as shown in Fig. 5(b). For traffic camera lives, we collect both day and night videos to capture varying video content. The videos are 30fps and vary from 360p to 1080p, and a weak device like NX has 360p video to ensure it can afford the inference. The results of a large model, YOLOv7-e6e, is regarded as the ground truth, which is consistent with previous works [39], [40]. We measure the inference energy cost γ_i for devices from NX to Orin as 5.42W, 13.35W, 17.92W, and 24.24W respectively. Each slot is set to 1s to ensure timely adaptation.



(a) Overall Accuracy (b) Latency Distribution

Fig. 6: Testbed Results on Accuracy and Latency



(a) Detection and Tracking (b) Number of Inference

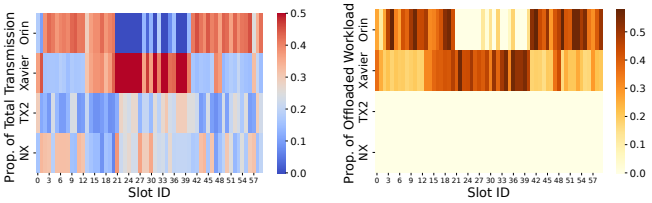
Fig. 7: Detailed Detection Results

Algorithms: Besides evaluating our method VisFlow, we compare it with algorithms as follows:

- **Local:** Each device separately conducts as many as possible inferences for its video stream in local.
- **Weak Device Priority (WeakPri):** Each strong device uses half the time to process its frames, and each weak device offloads as many frames as possible to the strongest devices successively, considering the device capability and the network status of the last slot.
- **Uniform Provisioning (UniProv):** Ensure that the video of each device is detected at the same frequency when offloading under constraints. If any device has spare computing resources after offloading, it detects for its own.

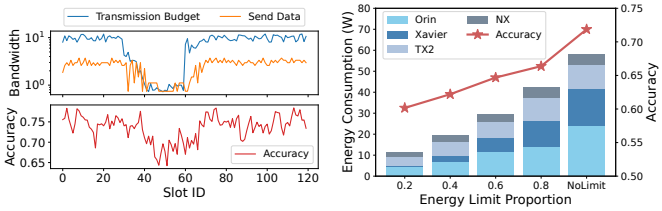
C. Testbed Results

Accuracy and Latency: First, we compare the overall accuracy and latency distribution of different methods. In Fig. 6(a), with proper frame inference offloading, VisFlow gets average accuracy of 0.718, while Local, WeakPri, and UniProv only get 0.607, 0.667 and 0.652 respectively. In addition, since VisFlow makes more detection for cameras with higher detection demands, the analysis quality is more robust with less low accuracy values and higher aggregation. In Fig. 6(b), we depict the end-to-end latency distribution of frames, i.e., the time between the frame and its result. Although VisFlow has an average latency slightly higher (about 18%) than WeakPri and UniProv due to more frame offloading and transmission, the real-time capability actually depends on the longest end-to-end latency of frames. To exclude the outliers, we use the 95th percentage of latency to measure the real-time performance. VisFlow has 95% frames getting results within 611ms, which means we get real-time analytics for 95% frames with a lag of 611ms. However, due to high inference latency or neglect of network fluctuation, the lags of Local, WeakPri and UniProv are 938ms, 912.2ms, and 677.3ms respectively.



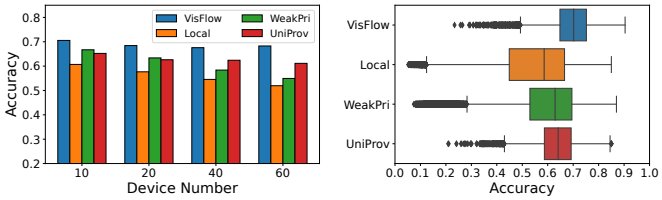
(a) Transmission Percentage (b) Workload from Offloading

Fig. 8: Workload Balancing with Network Fluctuation



(a) Adaptation to Transmission (b) Adaptation to Energy

Fig. 9: Adaptation to Transmission and Energy Limits



(a) Accuracy in Varying Scales (b) Accuracy Distribution

Fig. 10: Large-scale Verification of VisFlow

Detailed Detection Results: Then we explore the source of the accuracy difference between methods in detail. Fig. 7(a) illustrates that VisFlow's main improvement comes from the tracking accuracy. While our detection accuracy is only higher than other methods by about 3%, the tracking accuracy is higher than other methods by 36.3%, 9.3% and 11.9% respectively. It shows that VisFlow effectively decreases tracking drift and long-term tracking loss. Fig. 7(b) further depicts how VisFlow decreases tracking drift. It shows that VisFlow conducts 1,273 detections per minute on average, and is higher than others by 31.2%, 34.6% and 14.5% respectively. Further, VisFlow shows proper provisioning between devices. Contrarily, Local has 62% of inference performed for Orin due to uneven resources, WeakPri puts too much for the weakest device, and UniProv's balancing strategy is non-optimal.

Workload Balancing: Furthermore, we explore the process of workload balancing in VisFlow with network fluctuations. Fig. 8(a) shows the percentage of transferred data of each device compared to the total transmission in each slot, and Fig. 8(b) depicts how much of the device's workload is for other devices' inference tasks (for weak devices, the value is 0 since they don't receive tasks). It depicts that Orin receives most of transmission data and workload from offloading due to strong capability and offloading decisions in the beginning. In slots 13-40, the network of Orin fluctuates and the transmission budget decreases rapidly, thus VisFlow transforms the offloaded workloads from weak devices to Xavier as a substitution. After the recovery of Orin, the workload is mainly transferred back to Orin for lower inference latency.

Flexible Adaptation: Fig. 9 shows the flexible adaptive decisions regarding varying constraints in VisFlow. Fig. 9(a) depicts the transmission and accuracy variation of weak device NX when facing network congestion. As bandwidth decreases from 10 to 0.5, the scheduler also reduces the sending data size to fit the budget, which is promoted by the queue. As a result, NX has to conduct detections locally with high latency and long-term tracking, thus the accuracy decreases from 0.75 to 0.65 temporally. Further, the transmission and accuracy both recover after the network congestion is over. Fig. 9(b) shows the performance of VisFlow with varying energy consumption limits. As the energy limit proportion decreases from 1 (i.e., no limit) to 0.2, the energy consumption of Orin, Xavier, TX2 and NX reduces 81.9%, 96.7%, 60% and 62.2% respectively. Thus, with limited energy, the system should give priority to reducing the usage of strong devices which cost more energy.

Large-scale Verification: Finally, we verify the performance of VisFlow with large-scale cameras. In Fig. 10(a), we upscale the device number to 10, 20, 40 and 60 respectively and keep the proportion of Orin, Xavier, TX2 and NX number as 1:2:3:4. As the device number increases from 10 to 60, VisFlow increases the accuracy by up to 16.6%-25.9% compared with other methods. With a larger scale, VisFlow shows greater superiority due to proper inference provisioning regarding video content and network status compared with other methods. Fig. 10(b) takes the scene with device number 40 as an example to depict accuracy distribution for each method. It shows that VisFlow also decreases the tail accuracy and gets more aggregated high-accuracy results for all frames by jointly optimizing tracking and detection accuracy.

VI. Related Work

Detection-with-Tracking Framework: Considering limited camera resources, some works are based on "detection-with-tracking" framework to achieve real-time analytics. Glimpse [7] proposes to combine detection and tracking, and only trigger detection when there is a great deviation between consecutive frames. Liu *et al.* [41] propose to use motion vectors extracted from encoded frames to perform object tracking, which helps save time. Reducto [6] explores different video features to measure the deviation like edge, pixel and area, and select different features and thresholds for varying analytics tasks and video content. Hanyao *et al.* [16] adaptively select the best threshold to trigger edge-assisted detections. EdgeDuet [42] offloads small objects to edge servers with tile-level parallelism, and adopts single-object tracker KCF for other objects locally.

However, the works above are all assisted by edge servers, which can be influenced by fluctuating network environment and brings high network cost. In addition, they ignore to make full use of camera-side resources.

Video Analytics with Distributed Cameras: Some works investigate the video analytics system deployed with distributed cameras. LEVEA [17] offloads computation between edge nodes and clients using edge-first strategy, and formulated an optimization problem to minimize the response time.

Long *et al.* [18] study cooperative processing on mobile devices for delay-sensitive video processing tasks, and propose algorithms to map devices and video chunks to proper processing groups. SurveilEdge [19] proposes an intelligent task allocator to achieve a latency-accuracy tradeoff and balance the load among computing nodes, supporting large-scale surveillance video streams. Distream [20] designs a mechanism to adapt to the workload dynamics on the smart camera-edge cluster architecture, which smartly partitions workloads between cameras and the edge cluster.

And some works [43]–[46] also exploit and utilize spatial and temporal locality in distributed camera networks. RexCam [45] and Spatula [43] build a cross-camera correlation model based on historical traffic patterns and use it to filter frames or prune the search space of a query identity. CrossRoI [44] removes the repentant appearances of identical objects in multiple cameras without harming accuracy. Li *et al.* [46] enable co-located cameras with overlapping fields of view to share inference results, eliminating redundant inferences.

However, the works above fail to consider the tracking technique to assist video analytics on the camera side, and some of them still rely on edge servers to make inferences.

VII. Conclusion and Future Work

Live video analysis on collaborative heterogeneous cameras plays an important role in surveillance and public safety. In this paper, we propose a content-aware inference offloading mechanism to perform more detections for cameras with higher detection demands in a detection-with-tracking framework. We formulate the problem considering tracking drift, detection quality and limited resources. The problem maximizes the long-term overall accuracy under constrained computational, network and energy resources. The testbed on real-world videos confirms that our approach improves the accuracy by up to 18.3% compared to the alternatives.

Despite our work, several areas warrant further exploration. First, attaining real-time analytics on distributed cameras presents an intriguing challenge, for example, bandwidth allocation could be explored subsequently to orchestrate more effective transmissions. Second, some devices at edges could potentially support multiple models, which would afford greater flexibility in detection offloading and is a promising avenue for future investigation.

Appendix

A. Proof of Lemma 2

Proof. First, note that $\hat{\boldsymbol{x}}_t$ is a random variable vector with $\mathbb{E}[\hat{\boldsymbol{x}}_t] = \widetilde{\boldsymbol{x}}_t$. Using the definition in (15), we have

$$\begin{aligned}\mathrm{Reg}_T^{\mathrm{d}} &\overset{(27a)}{\leq} \mathbb{E}\left[\sum\nolimits_{t=1}^{T} f_t(\hat{\boldsymbol{x}}_t)\right] - \sum\nolimits_{t=1}^{T} f_t^G(\widetilde{\boldsymbol{x}}_t^*) \\ &\overset{(27b)}{\leq} \sum\nolimits_{t=1}^{T} f_t(\widetilde{\boldsymbol{x}}_t) + M\sigma_\beta - \sum\nolimits_{t=1}^{T} f_t^G(\widetilde{\boldsymbol{x}}_t^*) \\ &\overset{(27c)}{=} \widetilde{\mathrm{Reg}}_T^{\mathrm{d}} + M\sigma_\beta, \end{aligned} \tag{27}$$

where Inequality (27a) holds due to the real optimum solution is better than the integral optimum solution; Inequality (27b) utilizes the Jensen gap in [33] and $M\sigma_\beta$ is introduced as constant bound related to the variation of f_t; Equality (27c) holds using the definition in (16).

Using the definition in (17), we have $\forall k$,

$$\begin{aligned}\mathrm{Vio}_{T,k}^{\mathrm{d}} &= \mathbb{E}[\sum\nolimits_{t=1}^{T} g_{t,k}(\hat{\boldsymbol{x}}_t)] = \sum\nolimits_{t=1}^{T} \mathbb{E}\left[g_{t,k}(\hat{\boldsymbol{x}}_t)\right] \\ &\overset{(28a)}{=} \sum\nolimits_{t=1}^{T} g_{t,k}(\mathbb{E}\left[\hat{\boldsymbol{x}}_t\right]) = \sum\nolimits_{t=1}^{T} g_{t,k}(\widetilde{\boldsymbol{x}}_t) = \widetilde{\mathrm{Vio}}_{T,k}^{\mathrm{d}}, \end{aligned} \tag{28}$$

where Equality (28a) holds since $g_{t,k}$ is linear and the linearity of expectation. □

B. Proof of Theorem 1

Proof. **Bound $\mathbf{Reg}_T^{\mathbf{d}}$:** From the update strategy of the virtual queue, we have $\forall t, k, Q_{t+1,k} = \max\{-g_{t,k}(\boldsymbol{x}_{t+1}), Q_{t,k} + g_{t,k}(\boldsymbol{x}_{t+1})\}$. Thus, it is satisfied that $\forall t$,

$$\begin{aligned}\|\boldsymbol{Q}_{t+1}\|_2 &= (\sum\nolimits_{k=1}^{N+1} |Q_{t+1,k}|^2)^{\frac{1}{2}} \\ &\geq (\sum\nolimits_{k=1}^{N+1} |-g_{t+1,k}|^2)^{\frac{1}{2}} \geq \|\boldsymbol{g}_t(\widetilde{\boldsymbol{x}}_{t+1})\|_2 . \end{aligned} \tag{29}$$

According to [32], we obtain:

$$\begin{aligned}&\sum_{t=1}^{T} f_t(\widetilde{\boldsymbol{x}}_t) - \sum_{t=1}^{T} f_t(\widetilde{\boldsymbol{x}}_t^*) \leq \frac{1}{2}\|\boldsymbol{g}_T(\widetilde{\boldsymbol{x}}_{T+1})\|_2^2 - \frac{1}{2}\|\boldsymbol{Q}_{T+1}\|_2^2 \\ &+V_f + V_{\boldsymbol{g}} + 4\mu R V_{\boldsymbol{x}} + \|\boldsymbol{g}_1(\widetilde{\boldsymbol{x}}_1)\|_2^2 + F + f_1(\widetilde{\boldsymbol{x}}_1) + \|\widetilde{\boldsymbol{x}}_1 - \widetilde{\boldsymbol{x}}_1^*\|_2^2 \\ &\overset{(30a)}{\leq} V_f + V_{\boldsymbol{g}} + 4\mu R V_{\boldsymbol{x}} + C_0 \end{aligned} \tag{30}$$

where Inequality (30a) holds by applying (29) and representing the sum of constants by C_0. Then, we can derive that

$$\begin{aligned}\widetilde{\mathrm{Reg}}_T^{\mathrm{d}} &= \sum\nolimits_{t=1}^{T} \left[f_t(\widetilde{\boldsymbol{x}}_t) - f_t^G(\widetilde{\boldsymbol{x}}_t^*)\right] \\ &= \sum\nolimits_{t=1}^{T} \left[f_t(\widetilde{\boldsymbol{x}}_t) - f_t(\widetilde{\boldsymbol{x}}_t^*)\right] + \left[f_t(\widetilde{\boldsymbol{x}}_t^*) - f_t^G(\widetilde{\boldsymbol{x}}_t^*)\right] \\ &\overset{(31a)}{\leq} V_f + V_{\boldsymbol{g}} + 4\mu R V_{\boldsymbol{x}} + C_0 + \theta_0, \end{aligned} \tag{31}$$

where Inequality (31a) holds by plugging (30) and Assumption 3. Utilizing Lemma 2, we have

$$\mathrm{Reg}_T^{\mathrm{d}} \leq \widetilde{\mathrm{Reg}}_T^{\mathrm{d}} + M\sigma_\beta \leq V_f + V_{\boldsymbol{g}} + 4\mu R V_{\boldsymbol{x}} + C_1, \tag{32}$$

where $C_1 = C_0 + \theta_0 + M\sigma_\beta$ is a constant.

Bound $\mathbf{Vio}_T^{\mathbf{d}}$: According to [32], we have

$$\|\boldsymbol{Q}_T\|_2 \leq 4V_{\boldsymbol{\lambda}} + 2\sqrt{V_f} + \sqrt{2V_{\boldsymbol{g}}} + \sqrt{8\mu R V_{\boldsymbol{x}}} + C_0', \tag{33}$$

where $C_0' = \sqrt{G^2 + 4F + 2\mu\|\widetilde{\boldsymbol{x}}_1 - \widetilde{\boldsymbol{x}}_1^*\|_2^2 + 2\|\boldsymbol{g}_1(\widetilde{\boldsymbol{x}}_1)\|_2^2} + 2\|\boldsymbol{\lambda}_1^*\|_2$.

Then using Lemma 2, we obtain

$$\begin{aligned}\mathrm{Vio}_{T,k}^{\mathrm{d}} &= \widetilde{\mathrm{Vio}}_{T,k}^{\mathrm{d}} = \sum\nolimits_{t=1}^{T} g_{t,k}(\widetilde{\boldsymbol{x}}_t) \leq \sum\nolimits_{t=1}^{T-1} g_{t,k}(\widetilde{\boldsymbol{x}}_{t+1}) \\ &\quad + g_{1,k}(\widetilde{\boldsymbol{x}}_1) + \sum\nolimits_{t=1}^{T-1} |g_{t+1,k}(\widetilde{\boldsymbol{x}}_{t+1}) - g_{t,k}(\widetilde{\boldsymbol{x}}_{t+1})| \\ &\overset{(34a)}{\leq} g_{1,k}(\widetilde{\boldsymbol{x}}_1) + Q_{T,k} + \widetilde{V}_{\boldsymbol{g}} \leq g_{1,k}(\widetilde{\boldsymbol{x}}_1) + \|\boldsymbol{Q}_T\|_2 + \widetilde{V}_{\boldsymbol{g}} \\ &\overset{(34b)}{\leq} 4V_{\boldsymbol{\lambda}} + 2\sqrt{V_f} + \sqrt{2V_{\boldsymbol{g}}} + \sqrt{8\mu R V_{\boldsymbol{x}}} + C_2, \end{aligned} \tag{34}$$

where Inequality (34) holds due to the queue updating and plugging (22); Inequality (34) holds by applying (33) and defining $C_2 = C_0' + g_{1,k}(\widetilde{\boldsymbol{x}}_1)$ as an constant. □

REFERENCES

[1] G. Ananthanarayanan, P. Bahl, P. Bodík, K. Chintalapudi, M. Philipose, L. Ravindranath, and S. Sinha, "Real-time video analytics: The killer app for edge computing," *computer*, vol. 50, no. 10, pp. 58–67, 2017.

[2] Q. Zhang, H. Sun, X. Wu, and H. Zhong, "Edge video analytics for public safety: A review," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1675–1696, 2019.

[3] I. E. Olatunji and C.-H. Cheng, "Video analytics for visual surveillance and applications: An overview and survey," *MLP*, pp. 475–515, 2019.

[4] K. Du, A. Pervaiz, X. Yuan, A. Chowdhery, Q. Zhang, H. Hoffmann, and J. Jiang, "Server-driven video streaming for deep learning inference," in *SIGCOMM*, 2020, pp. 557–570.

[5] S. Jiang, Z. Lin, Y. Li, Y. Shu, and Y. Liu, "Flexible high-resolution object detection on edge devices with tunable latency," in *MobiCom*, 2021, pp. 559–572.

[6] Y. Li, A. Padmanabhan, P. Zhao, Y. Wang, G. H. Xu, and R. Netravali, "Reducto: On-camera filtering for resource-efficient real-time video analytics," in *SIGCOMM*, 2020, pp. 359–376.

[7] T. Y.-H. Chen, L. Ravindranath, S. Deng, P. Bahl, and H. Balakrishnan, "Glimpse: Continuous, real-time object recognition on mobile devices," in *SenSys*, 2015, pp. 155–168.

[8] H. Kiani Galoogahi, A. Fagg, and S. Lucey, "Learning background-aware correlation filters for visual tracking," in *ICCV*, 2017, pp. 1135–1143.

[9] Y. Xiang, A. Alahi, and S. Savarese, "Learning to track: Online multi-object tracking by decision making," in *ICCV*, 2015, pp. 4705–4713.

[10] K. Lebeda, S. Hadfield, J. Matas, and R. Bowden, "Long-term tracking through failure cases," in *ICCV Workshops*, 2013, pp. 153–160.

[11] K. Yang, J. Yi, K. Lee, and Y. Lee, "Flexpatch: Fast and accurate object detection for on-device high-resolution live video analytics," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 2022, pp. 1898–1907.

[12] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *CVPR*, 2017, pp. 2462–2470.

[13] Z. Xiao, Z. Xia, H. Zheng, B. Y. Zhao, and J. Jiang, "Towards performance clarity of edge video analytics," in *2021 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 2021, pp. 148–164.

[14] C. Wang, S. Zhang, Y. Chen, Z. Qian, J. Wu, and M. Xiao, "Joint configuration adaptation and bandwidth allocation for edge-based real-time video analytics," in *INFOCOM*. IEEE, 2020, pp. 257–266.

[15] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 7464–7475.

[16] M. Hanyao, Y. Jin, Z. Qian, S. Zhang, and S. Lu, "Edge-assisted online on-device object detection for real-time video analytics," in *INFOCOM*. IEEE, 2021, pp. 1–10.

[17] S. Yi, Z. Hao, Q. Zhang, Q. Zhang, W. Shi, and Q. Li, "Lavea: Latency-aware video analytics on edge computing platform," in *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, 2017, pp. 1–13.

[18] C. Long, Y. Cao, T. Jiang, and Q. Zhang, "Edge computing framework for cooperative video processing in multimedia iot systems," *IEEE Transactions on Multimedia*, vol. 20, no. 5, pp. 1126–1139, 2017.

[19] S. Wang, S. Yang, and C. Zhao, "Surveiledge: Real-time video query based on collaborative cloud-edge deep learning," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 2519–2528.

[20] X. Zeng, B. Fang, H. Shen, and M. Zhang, "Distream: scaling live video analytics with workload-adaptive distributed edge intelligence," in *SenSys*, 2020, pp. 409–421.

[21] J. Jiang, G. Ananthanarayanan, P. Bodik, S. Sen, and I. Stoica, "Chameleon: scalable adaptation of video analytics," in *SIGCOMM*, 2018, pp. 253–266.

[22] R. Bhardwaj, Z. Xia, G. Ananthanarayanan, J. Jiang, Y. Shu, N. Karianakis, K. Hsieh, P. Bahl, and I. Stoica, "Ekya: Continuous learning of video analytics models on edge compute servers," in *NSDI*, 2022, pp. 119–135.

[23] M. Xu, T. Xu, Y. Liu, and F. X. Lin, "Video analytics with zero-streaming cameras," in *ATC*. USENIX, 2021, pp. 459–472.

[24] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *TPAMI*, vol. 37, no. 3, pp. 583–596, 2014.

[25] "Kabukicho live channel ii," July. 2023. [Online]. Available: https://www.youtube.com/watch?v=gFRtAAmiFbE

[26] "Shinjuku kabukicho live camera," July. 2023. [Online]. Available: https://www.youtube.com/watch?v=EHkMjfMw7oU

[27] Y. Chen, S. Zhang, Y. Jin, Z. Qian, M. Xiao, W. Li, Y. Liang, and S. Lu, "Crowdsourcing upon learning: Energy-aware dispatch with guarantee for video analytics," *IEEE Transactions on Mobile Computing*, 2023.

[28] R. M. Karp, *Reducibility among combinatorial problems*. Springer, 2010.

[29] X. Ran, H. Chen, X. Zhu, Z. Liu, and J. Chen, "Deepdecision: A mobile deep learning framework for edge video analytics," in *IEEE INFOCOM 2018-IEEE conference on computer communications*. IEEE, 2018, pp. 1421–1429.

[30] A. Jadbabaie, A. Rakhlin, S. Shahrampour, and K. Sridharan, "Online Optimization : Competing with Dynamic Comparators," in *AISTATS*, G. Lebanon and S. V. N. Vishwanathan, Eds., vol. 38. PMLR, 09–12 May 2015, pp. 398–406.

[31] L. Zhang, T.-Y. Liu, and Z.-H. Zhou, "Adaptive regret of convex and smooth functions," in *ICML*. PMLR, 2019, pp. 7414–7423.

[32] X. Cao, J. Zhang, and H. V. Poor, "A Virtual-Queue-Based Algorithm for Constrained Online Convex Optimization With Applications to Data Center Resource Allocation," *JSTSP*, vol. 12, no. 4, pp. 703–716, 2018.

[33] X. Gao, M. Sitharam, and A. E. Roitberg, "Bounds on the jensen gap, and implications for mean-concentrated distributions," *arXiv preprint arXiv:1712.05267*, 2017.

[34] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *NIPS*, vol. 28, 2015.

[35] Y. Yan, Y. Jin, S. Zhang, F. Cheng, Z. Qian, and S. Lu, "Focus! provisioning attention-aware detection for real-time on-device video analytics," in *2022 19th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. IEEE, 2022, pp. 271–279.

[36] "Oxford," July. 2023. [Online]. Available: https://www.youtube.com/watch?v=St7aTfoIdYQ

[37] "Jackson hole wyoming usa town square live cam," July. 2023. [Online]. Available: https://www.youtube.com/watch?v=1EiC9bvVGnk

[38] X. Wang, X. Zhang, Y. Zhu, Y. Guo, X. Yuan, L. Xiang, Z. Wang, G. Ding, D. J. Brady, Q. Dai, and L. Fang, "Panda: A gigapixel-level human-centric video dataset," in *CVPR*. IEEE, 2020.

[39] B. Zhang, X. Jin, S. Ratnasamy, J. Wawrzynek, and E. A. Lee, "Awstream: Adaptive wide-area streaming analytics," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, 2018, pp. 236–252.

[40] K. Du, Q. Zhang, A. Arapin, H. Wang, Z. Xia, and J. Jiang, "Accmpeg: Optimizing video encoding for accurate video analytics," *Proceedings of Machine Learning and Systems*, vol. 4, pp. 450–466, 2022.

[41] L. Liu, H. Li, and M. Gruteser, "Edge assisted real-time object detection for mobile augmented reality," in *MobiCom*, 2019, pp. 1–16.

[42] X. Wang, Z. Yang, J. Wu, Y. Zhao, and Z. Zhou, "Edgeduet: Tiling small object detection for edge assisted autonomous mobile vision," in *INFOCOM*. IEEE, 2021, pp. 1–10.

[43] S. Jain, X. Zhang, Y. Zhou, G. Ananthanarayanan, J. Jiang, Y. Shu, P. Bahl, and J. Gonzalez, "Spatula: Efficient cross-camera video analytics on large camera networks," in *2020 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 2020, pp. 110–124.

[44] H. Guo, S. Yao, Z. Yang, Q. Zhou, and K. Nahrstedt, "Crossroi: Cross-camera region of interest optimization for efficient real time video analytics at scale," in *Proceedings of the 12th ACM Multimedia Systems Conference*, 2021, pp. 186–199.

[45] S. Jain, X. Zhang, Y. Zhou, G. Ananthanarayanan, J. Jiang, Y. Shu, and J. Gonzalez, "Rexcam: Resource-efficient, cross-camera video analytics at scale," *arXiv preprint arXiv:1811.01268*, 2018.

[46] J. Li, L. Liu, H. Xu, S. Wu, and C. J. Xue, "Cross-camera inference on the constrained edge," in *Proc. IEEE INFOCOM*, 2023.